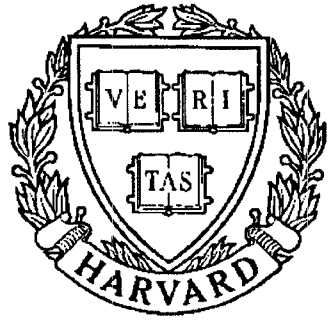
TECHNICAL RESEARCH REPORT

SYSTEMS RESEARCH CENTER



Supported by the National Science Foundation Engineering Research Center Program (NSFD CD 8803012), Industry and the University

Manufacturing Cell Design Using Simulated Annealing: an Industrial Application

by G. Harhalakis, J.M. Proth and X.L. Xie

TR 90-68

Journal of Intelligent Manufacturing (1990) **1** 185–191

Manufacturing cell design using simulated annealing: an industrial application

G. HARHALAKIS,[1] J. M. PROTH[2] and X. L. XIE[3]

[1] *Systems Research Center and Department of Mechanical Engineering, University of Maryland, USA*
[2] *INRIA-Lorraine, Project SAGEP, Cescom, Technopôle Metz 2000, 4, rue Marconi, Metz, France and Associate member of CIM-LAB of the Systems Research Center and Department of Mechanical Engineering, University of Maryland, USA*
[3] *INRIA-Lorraine, Project SAGEP, Cescom, Technopôle Metz 2000, 4, rue Marconi, Metz, France*

Received April 1990 and accepted May 1990

In this paper, we give a brief summary of simulated annealing (SA) procedures used to solve combinatorial optimization problems. We then present the manufacturing cell design problem which consists of designing cells of limited size in order to minimize inter-cell traffic. We show how to use a SA approach to obtain a good, if not optimum, solution to this problem. Finally, we apply this approach to an industrial problem and compare the results to the ones obtained using the so-called twofold heuristic algorithm.

Keywords: Group technology, clustering, simulated annealing, combinatorial problem, manufacturing cells

1. Introduction

The problem of partitioning a manufacturing system into cells has attracted the attention of researchers for a long time. Various approaches, involving more or less human expertise, are available (Askin and Subramnian, 1987; Garcia and Proth, 1986; McAuley, 1972; Garcia and Proth, 1985; Harhalakis, Nagi and Proth, 1990; King, 1979; Kumar, Kusiak and Vannelli, 1986; Kusiak, 1985). At one end of the spectrum, there are approaches which provide only an aid to human decision and do not take quantitative criteria into account. At the other end of the spectrum, we have approaches which aim at optimizing quantitative criteria. The approach presented in this paper belongs to the latter type. It aims at partitioning a manufacturing system into cells of limited size, the objective being to minimize the inter-cell traffic. This problem is NP-hard. Consequently, only heuristic or artificial intelligence based algorithms are able to provide a good, if not optimum, solution to this problem.

A twofold heuristic algorithm to solve this problem has already been proposed (Harhalakis, Nagi and Proth, 1990) and the industrial applications using this algorithm have provided very interesting results. Nevertheless, it seems that the well-known SA technique leads to results as good as previous ones and offer more advantages. In particular, SA can lead to many good solutions, i.e., to different solutions having low inter-cell traffic. It allows the designer to make his final choice amongst the set of good solutions on the basis of implicit or explicit, qualitative or quantitative criteria.

The paper is organized as follows. In Section 2, we present the basic concepts of SA and try to explain the advantages and the limitations of this kind of approach. Despite the fact that SA is known as being only a heuristic way to solve combinatorial optimization problems, it often leads to very good solutions and is currently being revisited by researchers (Johnson *et al.*, 1989). Section 3 is devoted to the application of SA to the problem on hand. The most important part of this section is the definition of the neighborhood of a set of manufacturing cells. Finally, Section 4 presents an industrial application of the SA approach and compares the results with those obtained when using the twofold algorithm proposed in (Harhalakis, Nagi and Proth, 1990).

2. Basic concepts of simulated annealing

In some sense, SA can be considered as an improvement of local optimization. Section 2.1 is therefore devoted to the specification of local optimization approaches. In Section 2.2, we introduce SA as an extension of local optimization and consider its physical basis. Finally, in Section 2.3, we provide some mathematical foundations of SA based on Markov chains.

2.1. *Local optimization*

Let us consider a combinatorial optimization problem P, S the set of its feasible solutions and an objective function f: $S \rightarrow R$ which associates a real value to each feasible solution $s \in S$. Remember that a feasible solution is a solution which verifies all the constraints of the problem. Solving P consists of finding $s^* \in S$ such that $f(s^*)$ is optimum, i.e.

$$f(s^*) = \underset{s \in S}{\text{opt}}\ f(s) \qquad (1)$$

In Equation 1, opt represents either the maximum or the minimum, depending on the problem.

Since P is a combinatorial problem, no exact procedure is usually available to solve it in a reasonable amount of time and one uses heuristic procedures to obtain a feasible solution which is expected to be near optimum. Local optimization procedures belong to the heuristic approaches. Such a procedure starts from a feasible solution $s \in S$. It then requires the computation of the neighborhood $H(s)$ of s. $H(s)$ is the set of solutions 'close' to s. They are obtained by perturbing s in an adequate manner (depending on P). If, for at least one $s' \in H(s)$, $f(s')$ is better than $f(s)$ according to Equation 1, one states $s = s'$ and restarts the procedure with the computation of $H(s)$. The process goes on until no $s' \in H(s)$ improves $f(s)$. The solution s is a local optimum. The problem with this kind of procedure is when this local optimum is very poor compared to the global optimum.

A local optimization procedure is uphill (when the goal is to maximize f) or downhill (when the goal is to minimize f). There is no way to avoid entrapment in the local optimum if it is poor (i.e. if the related value of f is far from the optimum). A way to partially overcome this difficulty is to restart the local optimization procedure many times with different initial feasible solutions, and to keep the best local optimum solution. But the difficulty is in the selection of the initial feasible solution, which, in some cases, could be very hard. Furthermore, the whole procedure has to be re-started for each trial, which usually leads to a result worse than the one obtained using a SA procedure for the same amount of computation.

2.2. *Simulated annealing*

In a local optimization procedure, one keeps a new solution s' belonging to the neighborhood $H(s)$ of s, if it improves the objective function. A SA procedure also allows one to keep s' with a given probability, if s' does not improve the objective function, and this probability decreases along with the amount of computation.

We owe this approach to Kirkpatrick *et al.* (Kirkpatrick, Gelatt and Vecchi, 1983; Darema *et al.*, 1987; Lundy and Mees, 1986; Vecchi and Kirkpatrick, 1983) whose work is based on the analogy of the behavior of a large number of atoms when the temperature decreases. At a high temperature, the energy of the set of atoms is high. When the temperature decreases, the energy of the system evolves depending on the cooling speed. If this speed is very high, the system reaches a 'frozen' state and the energy remains high. This situation is called 'chaos' and corresponds to the local optimum in a combinatorial problem. If the cooling speed is low, atoms move increasingly slowly, but they find their equilibrium state at each temperature and the energy is minimal when atoms reach their 'frozen' state: this state is called 'crystal'. The way to reach this state is explained in the SA optimization procedure introduced in this section.

The probability of the energy of the system being e at temperature T in the equilibrium state is given by Equation 2:

$$\Pr\{e/T\} = \exp\{-e/(K_B T)\}/U(T) \qquad (2)$$

where:

$$U(T) = \int_0^{+\infty} \exp\{-e/(K_B T)\}\,de$$

$U(T)$ is a standardization constant
K_B is Boltzmann's constant
T is the temperature

Here is a very simple algorithm (Metropolis *et al.*, 1953) to find an equilibrium state when the temperature T is known.

Algorithm 1

1. **Get an initial state s_0 and compute its energy e_0**
2. **Perturb s_0 in order to obtain s_1 belonging to the neighborhood of s_0 and compute the related energy e_1**
3. **Compute $\Delta_e = e_1 - e_0$**
4. **Test:**
   - 4.1. **If $\Delta_e \leq 0$, s_1 is accepted as a possible next state of the system**
   - 4.2. **If $\Delta_e > 0$, then s_1 is accepted as a possible next state of the system with the probability**

$$\Pr = \exp\{-\Delta_e/(K_B T)\}$$

5. **If s_1 is accepted as a possible state of the system, set $s_0 = s_1$**
6. **Go to 2**

This Monte Carlo simulation provides feasible equilibrium states of the system for temperature T after some computational steps. Combinational optimization uses a similar algorithm in conjunction with a procedure to reach

lower temperatures. In the previous explanation, let us replace state and energy, respectively, with the terms feasible solution and objective function value. Furthermore, let us introduce $r \in (0, 1)$ which is used to lower the temperature at each computational step. The algorithm to find a near optimum solution in the case of the minimization of the objective function is as follows (where ε is a small positive number provided by the user):

Algorithm 2

1. **Get an initial feasible solution s_0 and compute the related value of the objective function $f(s_0)$**
2. **Get an initial temperature $T > 0$**
3. **Generate a feasible solution s_1 in the neighborhood of s_0 and compute the related value of the objective function $f(s_1)$**
4. **Compute $\Delta_f = f(s_1) - f(s_0)$**
5. **Test:**
   - **5.1. If $\Delta_f \leq 0$, set $s_0 = s_1$**
   - **5.2. If $\Delta_f > 0$, set $s_0 = s_1$ with the probability exp $(-\Delta_f/T)$**
6. **Set $T = rT$**
7. **If $T > \varepsilon$, go to 3**
8. **Use local optimization to reach a local optimum starting from the last s_0 value**

Note that the last step is often neglected because local optimization does not significantly improve the results when ε is small. We finally keep the best solution among all those that have been obtained. When it comes to maximizing an objective function, just change $\Delta f \leq 0$ and $\Delta f > 0$ into $\Delta f \geq 0$ and $\Delta f < 0$ respectively. It is easily understandable that the difficulty in applying Algorithm 2 is usually the definition of the initial feasible solution, the initial temperature, parameter ε and the coefficient $r \in (0, 1)$. Note that Algorithm 2 is derived from earlier work (Kirkpatrick, Gelatt and Vecchi, 1983).

2.2.1. *Remarks*

(i) In Algorithm 2, the temperature decreases at each step of the computation. This algorithm is said to be heterogeneous by analogy with the Markov chains introduced in the next section. Another strategy consists in lowering the temperature every n steps of the computation. In that case, the algorithm is said to be piecewise homogeneous because it can be expressed as a sequence of finite homogeneous Markov chains. (ii) As we said above, T, ε and r are control parameters which have to be defined for each problem. (iii) The neighborhood of a given solution s_0 is the set of solutions obtained by applying an elementary perturbation to s_0. The concept of elementary disturbance depends on the problem on hand. Assume, for instance, that the goal of the combinatorial problem consists in finding an optimum order for a set of entities. Given an order s_0, we can define the neighborhood of s_0 as the set of orders obtained by reversing the order of two entities of s_0 and/or by changing the rank of one of the entities. Assume now that we aim to find an optimum partition of a set of entities. Given a partition s_0, an element of the neighborhood can be the partition derived from s_0 by putting an element belonging to a subset in another subset and/or by interchanging two entities located in different subsets. (iv) In the following, we assume that a state can be reached starting from any other state by applying a sequence of elementary disturbances. This is the case dealt with in this paper.

2.3. *Mathematical foundations*

2.3.1. *Markov chains*

A process which is evolving in time, in a manner controlled by a probabilistic law is a stochastic process. Let $E(t)$ be the state of such a process at time t. If the state of the process can evolve only at discrete points in time (for instance at times $t = 0, 1, 2, 3, \ldots$), the process is said to be a discrete parameter process. If the number of possible states is finite or countable infinite, the stochastic process is called a chain.

Let us denote the possible states of the chain by $1, 2, 3, 4, \ldots, n, n+1, \ldots$. A chain is a Markov chain if, the probability that the state of the process be i at time $k+1$ only depends on the state at time k (and not on the states of the process at times $0, 1, 2, \ldots, k-1$). In other words:

$$\begin{aligned} \Pr\{E(k+1) = i / E(k) = j_k \text{ and } E(k-1) \\ = j_{k-1} \text{ and } \ldots \text{ and } E(0) = j_0\} \\ = \Pr\{E(k+1) = i / E(k) = j_k\} \end{aligned} \tag{3}$$

For a pair of points in time (r, k) with $0 \leq r \leq k$ and a pair of states i, j, the conditional probability:

$$\Pr\{E(k) = i / E(r) = j\} = \Pr_{i,j}(r, k) \tag{4}$$

is called a transition probability. When the transition probabilities $\Pr_{i,j}(r, k)$ only depend on $k - r$, the probabilities are said to be stationary and the related Markov chain is called homogeneous.

2.3.2. *Simulated annealing*

We refer to a problem P with a finite number N of states, which is the case in the problem examined in this paper. The SA procedure is a Markov chain since, (i) the number of states is finite and, (ii) the probability $\Pr(s_0, s_1)$ to reach state s_1 just after s_0 only depends on s_0 (and not on the states preceding s_0). This Markov chain is non-homogeneous. Stating $\Delta(s_0, s_1) = f(s_1) - f(s_0)$, the probability $\Pr(s_0, s_1)$ is given by:

$$\Pr(s_0, s_1) = \begin{cases} 0 \text{ if } s_1 \notin H(s_0) \\ a \text{ if } s_1 \in H(s_0) \text{ and } \Delta(s_0, s_1) \leq 0 \\ a.\ \exp[-\Delta(s_0, s_1)/T_0] \text{ if } s_1 \in H(s_0) \text{ and } \Delta(s_0, s_1) > 0 \end{cases} \tag{5}$$

where a is such that $\Sigma_{s_i \in H(s_0)} \Pr(s_0, s_1) = 1$ and T_0 is the value of the temperature at state s_0. We assume, of course, that s_1 is chosen at random in $H(s_0)$.

As a consequence of Remark (iv) in Section 2.2 we know that, whatever the pair (s_0, s^*) of states, it is always possible to find an integer value v and a sequence $s_1, s_2, \ldots, s_v$ of states such that $s_i \in H(s_{i-1})$ for $i = 1, 3, \ldots, v+1$ (setting $s^* = s_{v+1}$).

In the following, $x+1$ is the length of the path $\gamma = (s_0, s_1, \ldots, s_v, s^*)$. γ is called the elementary path if it does not contain the same state twice. The probability of reaching s^* starting from s_0 using a path (elementary or not) of maximum length $v+1$ is given by Equation 6:

$$Q_{v+1}(s_0, s^*) = \sum_{x=1}^{v+1} \left\{ \sum_{\gamma \in \Gamma_x(s_0, s^*)} \left[\sum_{i=1}^{x} \Pr(s_{i-1}, s_i) \right] \right\} \quad (6)$$

where $s_{i+1} = s^*$ and $\Gamma_{x+1}(s_0, s^*)$ is the set of paths of length $x+1$ joining s_0 to s^* and such that $s_i \neq s^*$ when $i = 0, 1, \ldots, x$. We know, from Remark (iv) in Section 2.2, that at least one of the Γs is non-empty for $x \in \{1, 2, 3, \ldots, n, \ldots\}$. In Equation 6, if $\Gamma_{x+1}(s_0, s^*)$ is empty for $x \leq v+1$, then $Q_{v+1}(s_0, s^*) = 0$. Let $(s_1^{**}, s_2^{**}, \ldots, s_K^{**})$ be the local optima to problem P and $E_1, E_2, \ldots, E_K$ the sets of feasible solutions to problem P such that, if $s \in E_k (k \in \{1, 2, \ldots, K\})$, we reach s_k^{**} starting from s by using a local optimization algorithm as defined in Section 2.1. We also define $F_{v+1}^k(s_0)$ as the set of states s belonging to E_k, such that $\Gamma_{v+1}(s_0, s) \neq \phi$ $(k = 1, 2, \ldots, K)$. In other words, $F_{v+1}^k(s_0)$ is the set of states, s^*, of E_k such that there exists at least one path of length $v+1$ joining s_0 to s^* and such that $s_i \neq s^*$ when $i = 0, 1, \ldots, v$. According to Algorithm 2, the length w of the path generated by the SA algorithm when the parameters are r, T and ε is such that $r^w T > \varepsilon \geq r^{w+1} T$.

Thus w is the only integer verifying:

$$\ln(\varepsilon/T)/\ln r - 1 \leq w < \ln(\varepsilon/T)/\ln r$$

and the probability of reaching the local optimum s_k^* starting from the initial state s_0 is:

$$R(s_k^*/s_0) = \sum_{s \in F_w^k} Q_w(s_0, s) + \sum_{x=1}^{w-1} Q_x(s_0, s_k^{**}) \quad (7)$$

The first term provides the probability to reach a state of E_k with a path length of w. The second term is the probability to reach s_k^{**} with a path length $\leq w-1$.

We assume that a local optimization follows the state reached using SA, i.e., that point 8 of Algorithm 2 holds, if it does not Equation 7 becomes

$$R(s_k^{**}/s_0) = \sum_{x=1}^{w} Q_x(s_0, s_k^{**}) \quad (7')$$

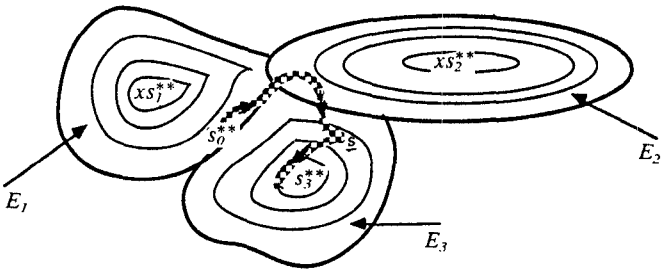


Fig. 1. A path generated using Algorithm 2

Of course, these probabilities strongly depend on the parameters ε, T and r. The smaller the ratio ε/T with regard to r, the higher the value of w and, as a consequence, the higher the probability that all the F_w^k $(k = 1, 2, \ldots, K)$ be non-empty. Thus, the smaller the ratio ε/T with regard to r, the higher the probability to reach the optimum value to any problem P, by making a large number of trials, but the larger the amount of computations for each trial. Figure 1 illustrates a SA process. In Fig. 1, s is the state reached when T becomes $\leq \varepsilon$. Path s to s_3^{**} is obtained by applying a local optimization procedure. The heavy lines show the limits of the sets E_k.

3. Manufacturing cell formation

The problem on hand is to group the machines into cells, in order to minimize the inter-cell traffic, assuming that the number of machines in each cell is limited (otherwise, the solution consisting of grouping all the machines in the same cell would be optimum). We state the problem in Section 3.1. In Section 3.2, we specify the neighborhood $H(s)$ of a partition s, of the set of machines into cells and we define the SA approach used for solving this problem. Finally, we present in Section 3.3 the application of this approach to an industrial problem and compare the results to those obtained by using the algorithm presented by Harhalakis and colleagues (Harhalakis, Nagi and Proth, 1990).

3.1. *Stating the problem*

Consider a set $M = \{M_1, M_2, \ldots, M_m\}$ of m machines and a set $J = \{J_1, J_2, \ldots, J_n\}$ of n part types. A unique routing, also called J_i, corresponds to each part type J_i $(i = 1, 2, \ldots, n)$. It identifies both the machines that are used for the manufacture and the sequence of operations performed. Thus $J_i = \{M_{i,1}, M_{i,2}, \ldots, M_{i,k_i}\}$ where $M_{i,j} \in M$ for $j = 1, 2, \ldots, k_i$. Let u_i be the weight of part J_i. The weight of a part type represents the volume of production over a large period (the same for all product types) or, similarly, the average production per time unit. We also introduce c_i $(i = 1, \ldots, n)$ which is the inter-cell transportation cost of

one unit product of type J_i. Note that the transportation cost could also relate a pallet load (i.e. a set of units of product). In this paper, c_i refers to one unit of product J_i. For each $(J_k, M_i, M_j) \in J \times M \times M$, we denote by q_{kij} the number of times M_j follows M_i or M_i follows M_j in the routing J_k. Then, for each pair $(M_i, M_j) \in M \times M$, the traffic between M_i and M_j is defined as follows:

$$t_{ij} = \sum_{k=1}^{n} c_k u_k q_{kij} \tag{8}$$

t_{ij} represents the transportation cost between M_i and M_j. We also set $t_{ii} = 0$ for $i = 1, 2, \ldots, m$.

Let $C = \{C_1, C_2, \ldots, C_q\}$ be a partition of M, i.e.:

$C_i \cap C_j = \phi$ for $i, j \in \{1, 2, \ldots, q\}$ and $i \neq j$

and

$$\bigcup_{i=1}^{q} C_i = M$$

We denote by $F(C)$ the total inter-cell traffic cost related to partition C. According to Equation 8, that cost can be expressed as follows:

$$\mathrm{F}(C) = \sum_{(i,j) \in E(C)} t_{ij} \tag{9}$$

where:

$$E(C) = \{(i,j)/i \in C_k j \in C_r, k, r \in \{1, 2, \ldots, q\}, k \neq r\}$$

Note that the number of pairs (i, j) in $E(C)$ is:

$$N(C) = q(q-1)/2$$

We also assume that the number of machines in a cell C_i $(i = 1, 2, \ldots, q)$ has an upper limit of N (integer):

$$\mathrm{card}(C_i) \leq N \text{ for } i = 1, 2, \ldots, q \tag{10}$$

Thus, the problem to solve is the following:

find a partition $C^* \in U$ of M such that:

$$\mathrm{F}(C^*) = \min_{C \in U} \mathrm{F}(C) \tag{11}$$

U being the set of partitions of M verifying Constraint 10.

3.2. *Problem solving*

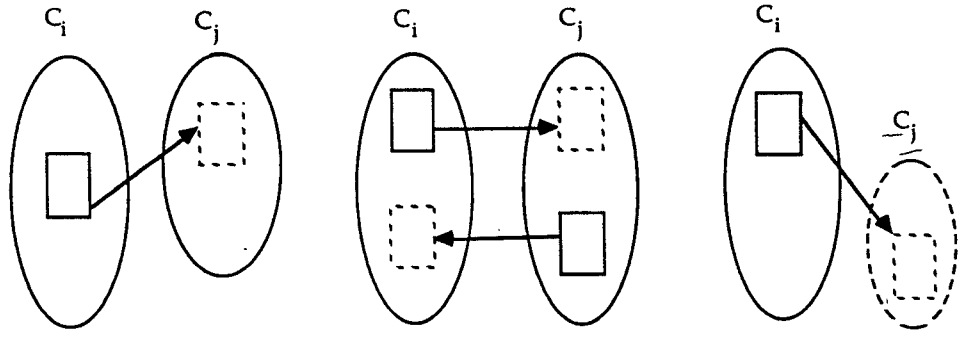
We first define the neighborhood of a partition. Given a partition C of M which verifies Constraint 10, the neighborhood $\mathrm{H}(C)$ of C is the set of partitions of U derived from C by either:

(i) Removing a machine from one cell to another cell.
(ii) Exchanging two machines located in two different cells.
(iii) Taking out a machine from a cell and generating a new cell with that machine.

Figure 2 summarizes the possible transformations to build a partition C' belonging to the neighborhood of C. C' belongs to the neighborhood of C if C' is obtained by applying to C one of the transformations (i), (ii) or (iii). The following result holds: Whatever C_0, $C^* \in U$ (i.e. partitions of M verifying Constraint 10), it is possible to find v and a sequence $C_1, C_2, \ldots, C_v$ of elements of U such that C_{i+1} belongs to the neighborhood of C_i for $i = 1, 2, \ldots, v$ (we set $C^* = C_{v+1}$). In other words, it is always possible to find a path joining C_0 to C^*.

3.2.1. Proof

C_0 to C^* can be easily joined up in a twofold process: (a) Starting from partition C_0, we first build C' the set of m partitions containing one machine each. That partition is reached by passing through a sequence $C_1, C_2, \ldots, C_{w-1}$, $C_w = C'$, C_{i+1} being derived from C_i $(i = 0, 1, \ldots, w-1)$



(i) Removing a machine from a cell and placing it to another cell

(ii) Interchanging two machines belonging to different cells

(iii) Generating a new cell

Fig. 2. Design of an element in the neighborhood

by using transformation (iii) (i.e. by generating a new cell). (b) We then derive c^* from $C' = C_w$ by building a sequence $C' = C_w, C_{w+1}, C_{w+2}, \ldots, C_{v-1}, C_v, C_{v+1} = C^*$ using transformation (i) to derive C_{i+1} from C_i ($i = w, w+1, \ldots, v$).

Q.E.D.

The previous result shows that it is always possible to join two partitions of U and the proof provides one of these paths. But this path is usually not unique. In the example given in the next section, we use Algorithm 2 after a slight modification by lowering the temperature only every L loops.

3.3. *An industrial example*

The example presented in this section has been solved for a French company whose goal was to reorganize one of its large shops. We had to handle 292 machines manufacturing 460 types of products. Parameters c_i and u_i ($i = 1, 2, \ldots, 460$) were equal to unity. The results obtained were compared to the ones obtained using the algorithm proposed by Harhalakis and colleagues (Harhalakis, Nagi and Proth, 1990). We first present that algorithm.

3.3.1. *The twofold heuristic algorithm*

This algorithm is a twofold algorithm composed of a bottom-up aggregation procedure followed by a local refinement procedure.

(i) *The bottom-up aggregation procedure*

This bottom-up aggregation procedure calls for the definition of the normalized inter-cell traffic T_{ij} between each pair (C_i, C_j) of cells:

$$T_{ij} = \left(\sum_{r/M_r \in C_i} \sum_{u/M_u \in C_j} t_{u,r} \right) / (n_i + n_j)$$

where n_i and n_j are respectively the number of machines in C_i and C_j. At the beginning of the algorithm, each machine is placed in a separate cell. Then, at each step of the aggregation procedure, we compute the normalized inter-cell traffic between each pair of cells and aggregate the two cells having the greatest inter-cell traffic, such that the total number of machines in the aggregate cell is smaller than N. The procedure is continued until it is either not possible to form any feasible aggregate cell due to the cell size limit constraint, or the traffic between each of the existing classes is zero.

(ii) *The local refinement procedure*

The local refinement procedure starts from the set of cells obtained at the end of the bottom-up aggregation procedure. Then, at each step of the procedure, a machine is considered as an external entity and the normalized inter-cell traffic between this machine and each of the cells is computed. The machine is assigned to the cell with which the traffic is maximal and whose size is $\leq N - 1$.

3.3.2. *Numerical results*

The results of the computation are summarized in Table 1 for various values of N. SA has been applied five times for each value of N, starting from the same initial feasible solution generated at random. The values used for the SA algorithm are $T = 500$ (initial temperature), $\varepsilon = 0.01$, $r = 0.95$ and $L = 30$ (number of consecutive loops in Algorithm 2 with the same temperature). Furthermore, we choose element (iii) in the neighborhood (Fig. 2) with the probability $1/(n_c + 1)$, where n_c is the number of cells in the current state. Each of the other two elements is chosen with probability $[1 - 1/(n_c + 1)]/2$.

As seen above, the SA procedure provides, on average, slightly better results than the twofold heuristic procedure if it is applied five times.

4. Conclusion

SA has been applied to numerous examples of various sizes. On average, it leads to better results than the ones obtained using the twofold heuristic procedure outlined in Section 3.3.1. A very important property of the SA approach is that it is possible to reach various good results starting from the same initial feasible solution. This property is of

Table 1. Numerical results.

	The twofold heuristic procedure			*SA procedure*		
N	*Number of cells*	*Inter-cell traffic*	*Computation time (s)*	*Number of cells*	*Inter-cell traffic*	*Computation time (s)*
5	59	47 470	981	59	48 160	386
				59	48 450	373
				59	48 829	384
				59	48 496	390
				59	49 140	370
10	30	42 607	1127	30	40 830	441
				31	43 537	402
				32	40 609	402
				31	41 707	428
				31	41 842	422
15	20	38 345	1215	22	37 591	450
				21	36 014	446
				21	37 612	443
				21	36 728	422
				21	38 011	465
20	15	36 746	1193	16	33 670	451
				17	33 561	459
				16	35 640	629
				16	33 545	444
				16	33 733	484

great interest when a feasible solution is difficult to obtain. It also enables us to propose to the designer partitions (i.e. sets of manufacturing cells) which are totally different but have similar inter-cell traffic values. The designer can then choose among good solutions (i.e. solutions with a small inter-cell traffic value), those which satisfy some supplementary qualitative criteria resulting from his/her expertise. There is, unfortunately, no procedure to obtain the best values for the parameters T (initial temperature) ε, r and L controlling the SA procedure: the selection of these values seem to be only a matter of experimentation.

References

Askin, R. and Subramnian, S. B. (1987) A cost-based heuristic for group technology configuration. *International Journal of Production Research*, **25** (1) 101–13.

Darema, F., Kirkpatrick, S. and Norton, V. A. (1987) Parallel algorithms for chip placement by simulated annealing. IBM *Journal Res. Development* **31**, 391–402.

Garcia, H. and Proth, J. M. (1985) Group technology in production management: the short horizon planning level. *Applied Stochastic Models and Data Analysis*, **1**, 25–34.

Garcia, H. and Proth, J. M. (1986) A new cross-decomposition algorithm: the GPM. Comparison with the Bond Energy Method. *Control and Cybernetics*, **15**, 115–65.

Harhalakis, G., Nagi, R. and Proth, J. M. (1990) An efficient heuristic in manufacturing cell formation for group technology applications. *International Journal of Production Research*, **28**, 185–98.

Johnson, D. S., Aragon, C. R., McGeoch, L. A. and Schevon, C. (1989) Optimization by simulated annealing: an experimental evaluation; Part 1, Graph partioning. *Operations Research*, **37**, 865–92.

King, J. R. (1979) Machine-component group formation in group technology. *OMEGA The International Journal of Management Science*, **8** (2) 193–9.

Kirkpatrick, S., Gelatt, C. D. and Vecchi, M. P. (1983) Optimization by simulated annealing. *Science*, **220**, 13 May.

Kumar, R. K., Kusiak, A. and Vannelli, A. (1986) Grouping of parts and components in flexible manufacturing systems. *European Journal of Operations Research*, **24**, 387–97.

Kusiak, A. (1985) The part families problem in flexible manufacturing systems. *Annals of Operations Research*, **3**, 279–300.

Lundy, M. and Mees, A. (1986) Convergence of an annealing algorithm. *Mathematical Programming*, **34** 111–24.

McAuley, J. (1972) Machine grouping for efficient production. *The Production Engineer*, **Feb.**, 53–7.

McCormick, W. T., Schweitzer, P. J. and White, T. E. (1972) Problem decomposition and data reorganization by a cluster technique. *Operations Research*, **20**.

Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A. and Teller, E. (1953) Equation of state calculations by fast computing machine. *J. Chem. Phys.*, **21**, 1087–92.

Vecchi, M. P. and Kirkpatrick, S. (1983) Global wiring by simulated annealing. *IEEE Trans. on Computer-Aided Design*, **CAD-2**, 215–22 (October).